**ICR Competition Records**

Pu Tan

Competition link: https://www.kaggle.com/competitions/icr-identify-age-related-conditions

Goal of the Competition

The goal of this competition is to predict if a person has any of three medical conditions. You are being asked to predict if the person has one or more of any of the three medical conditions (Class 1), or none of the three medical conditions (Class 0). You will create a model trained on measurements of health characteristics. To determine if someone has these medical conditions requires a long and intrusive process to collect information from patients. With predictive models, we can shorten this process and keep patient details private by collecting key characteristics relative to the conditions, then encoding these characteristics.

Goal: Discover the relationship between measurements of certain characteristics and potential patient conditions.

This documentation outlines my approach during the competition. The details may differ slightly from the actual implementation, with the full code available here [link].

The feature engineering techniques are described first, highlighting the key steps taken to construct informative features from the data. This includes [list main feature engineering techniques used].

Additional aspects of the methodology are then covered, including data preprocessing, hyperparameter tuning strategies, model evaluation metrics, and other considerations for this competition. Dilution records and other supplementary material are referenced in context where applicable.

While the documentation summarizes the techniques at a high-level, the full code contains the specifics for reproducibility. The goal is to provide an overview of the methodological process, with comprehensive details in the codebase. Suggestions to enhance formality and clarity of this competition writeup are welcomed.

Final model:

**lgbm cv 0.17 lb 0.16**

Tabpfn cv 0.23 lb

Xgb3 cv 0.17-0.19 lb

**NN**

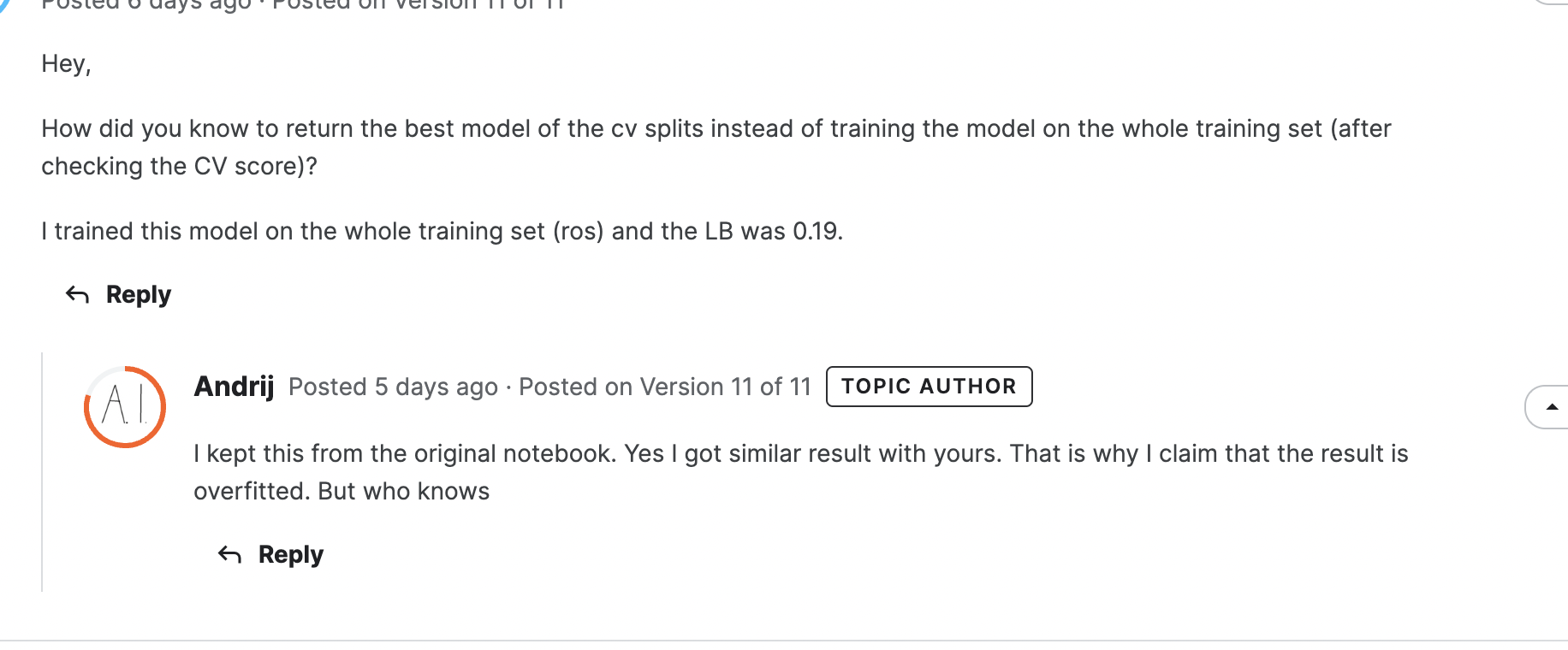
1、tabnet

|  |  |  |  |
| --- | --- | --- | --- |
| date | model | cv | figure |
| 0730 | tabnet | 0.2478 |  |
|  | wide&deep | 0.3241 |  |
|  | wide&deep+date | 0.3231 |  |
|  | wide&deep+DU binning | **0.2676** |  |
|  | wide&deep+DU binning+check BQ null | 0.2541 |  |
|  |  |  |  |

**Issue**

1. Currently, whether it's XGBoost or TabNet, the loss during training is AUC, not our target metric of balanced log loss. Will this have an impact?

2.



1. fitting with all date or other
2. divide into n folds and choose the model that fits best on those n folds to be the final model
3. TBD: looks like fitting based on the last fold each time

**Loss function**

**Balanced Log Loss Explained**

https://www.kaggle.com/competitions/icr-identify-age-related-conditions/discussion/422442

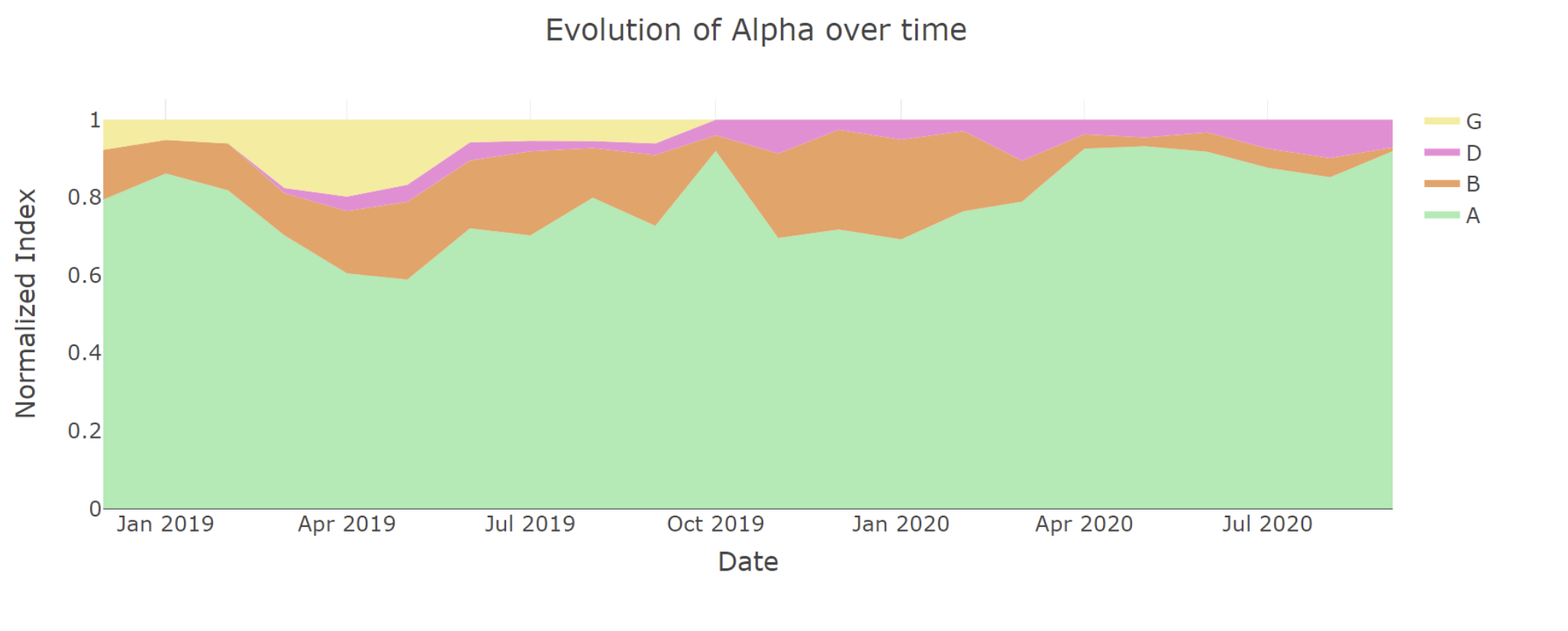
Many discussion posts and many notebooks (including all the highest scoring public notebooks with LB = 0.06) use an incorrect balanced log loss formula

**Greeks data**

For dates labeled as 'Unknown,' the Class is always 0.



2、As time passes, both G and B are decreasing. Why adding time to the test set would improve lb & cv.



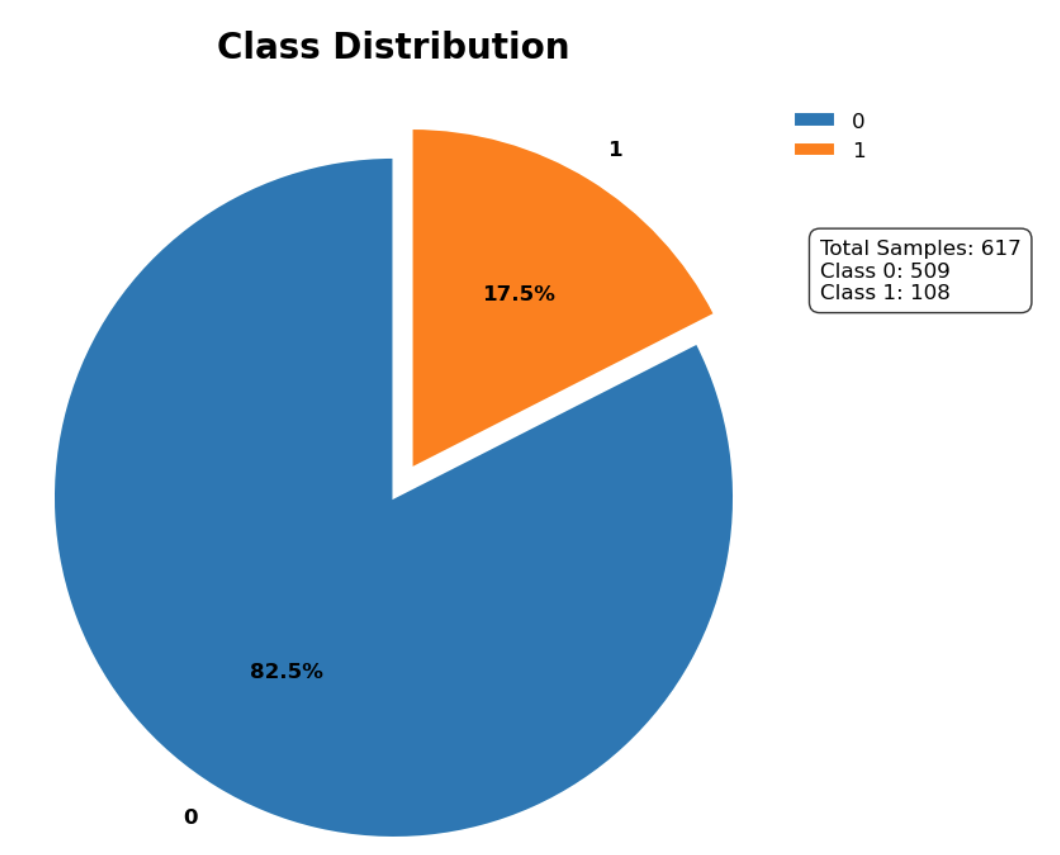
**greeks.Alpha and class**

Alpha as a label for stratification, predicting class. ❌

Use alpha as a label for stratification, predict alpha, and calculate the accuracy of the class

**EDA feature engineer**

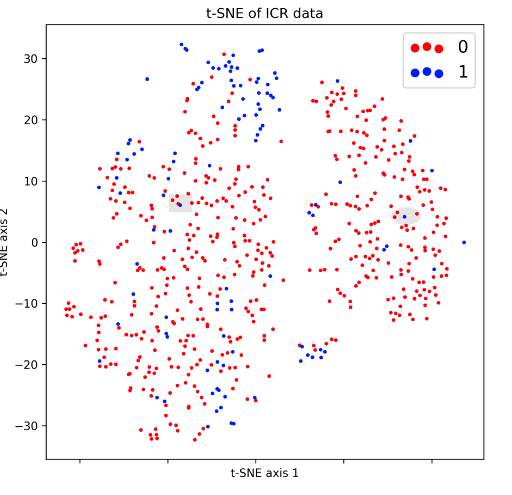
1、class distribution



2、feature distribution

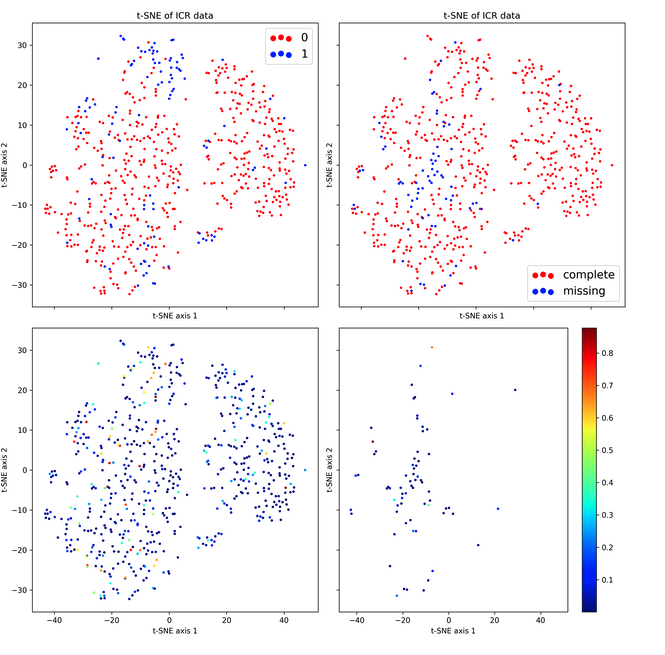
t-SNE to reduces high-dimensional data to two or three dimensions. (note: t-SNE focuses more on retaining the local features of the original data. Points that are close together in high-dimensional space are projected to be close together in the low-dimensional space as well.)

The distributions of the positive and negative classes are shown. For class 1, some points have most of their neighbors as class 0. These points are hard to predict, especially the blue points in the left rectangle in the bottom figure, which almost overlap with a red point. Their 56 features are almost the same.



Lower Left: Out-of-fold validation is performed on the training data to observe the error rate in predicting the true class labels. It shows that most of the predictions are fairly accurate (indicated in blue).

Lower Right: Focusing only on the points with missing values, the predictions are mostly good.



The idea is to identify points that are difficult to predict and then create a model to predict whether a point will be hard to forecast.

[useful idea](https://www.kaggle.com/competitions/icr-identify-age-related-conditions/discussion/421993)

[code](https://www.kaggle.com/code/cody11null/predicting-hard-cases)

Some say that by only considering relevant variables, the different classes in this graph can become more distinct.

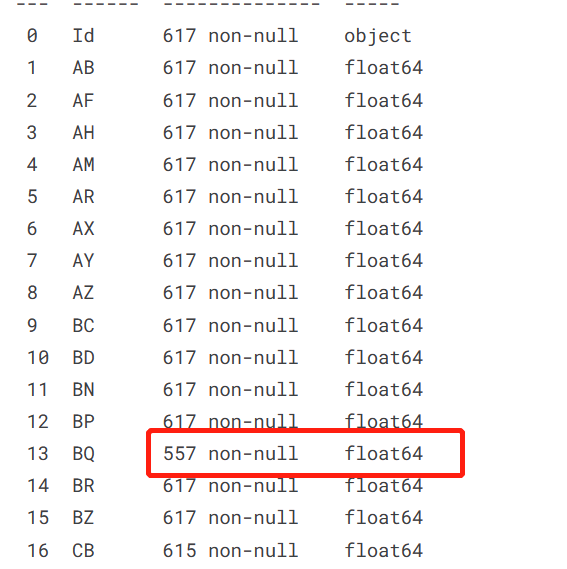
Pertains to the distribution of different categories as well as the distribution of points with missing values.

**Data Processing**

**Missing Value**

Two rows have many missing values; most such rows also have outliers. Simple mean or median imputation could be problematic, especially for rows with multiple outliers.

Note on BQ: Using a simple mean or median to fill in those 60 values could negatively impact the results, especially in rows with multiple outliers.



Methods

1. **KNNImputer https://blog.csdn.net/tMb8Z9Vdm66wH68VX1/article/details/130177587**

|  |
| --- |
| Python imp = KNNImputer() labels = train["Class"]  train = train.drop(columns="Class")  data = imp.fit\_transform(train) tmp = pd.DataFrame(columns=train.columns, data=data) train = pd.concat([tmp, labels], axis=1)train |

1. Calculate the correlation between the imputed columns and the class label, and choose different imputation methods accordingly.

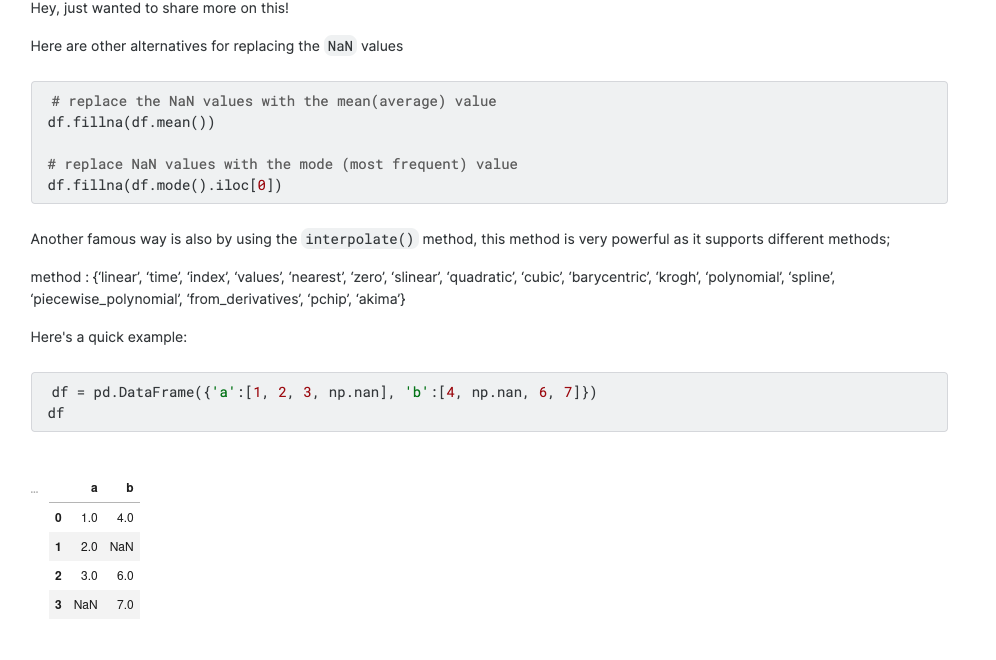
Filling with value 1.331155 changes correlation, meaning increases precision of outcome, - by 3%

|  |
| --- |
| Python df\_experimental = YOURDATASETNAME.loc[:,['BQ','EL' ,'Class']].copy()  df\_experimental["BQ\_NEW"]=df\_experimental['BQ'].fillna(ds["BQ"].mean()) ds["BQ\_NEW"]=np.where(ds['BQ']>0,ds['BQ'],1.331155)  df\_experimental["BQ\_NEW"].corr(df\_experimental["Class"]) ds["BQ\_NEW"].corr(ds["Class"]) |

2. Separate the data by class, use k-nearest neighbors (KNN) to impute missing values, and then combine the data back together.

2.1. Directly set to -1 and bucketize; missing values form their own separate category.

3. A missing value handling method mentioned in the comments section [Kaggle Discussion](<https://www.kaggle.com/competitions/icr-identify-age-related-conditions/discussion/410843>).



**Normalizations**

1. StandardScaler (Mean = 0, Standard Deviation = 1)

- Method: Subtract the mean and divide by the standard deviation. The transformed data follows a standard normal distribution with a mean of 0 and a standard deviation of 1.

- Transformation Function: x = (x-mean) / std

- Applicability: Suitable for data that already follows a normal distribution.

- Impact of Outliers: Somewhat robust to outliers, although outliers still affect the calculation of mean and standard deviation.

2. MinMaxScaler (0-1 Scaling)

- Method: Scale the features to fall within a given minimum and maximum value range, typically between 0 and 1. This is a linear transformation of the original data.

- Transformation Function: x = (x-min) / (max-min)

- Applicability: Suitable for data with a relatively stable range. If new data points alter the max/min values, rescaling will be necessary.

- Impact of Outliers: Highly sensitive to outliers, as they can distort the minimum and maximum values used for scaling.

3. RobustScaler (Quantile Scaling)

- Method: This scaler removes the median and scales the data based on the Interquartile Range (IQR). The IQR is the range between the 1st quartile (25th percentile) and the 3rd quartile (75th percentile).

- Applicability: Suitable for data that contains many outliers.

- Impact of Outliers: The RobustScaler minimizes the impact of outliers by scaling using the IQR, thereby making it robust to outliers.

By using the IQR for scaling, the RobustScaler is less influenced by extreme values and can be particularly useful when the dataset contains many outliers. This method is often recommended for data sets where the features are not normally distributed and contain a large number of outlier values.

[refer](https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/)

|  |
| --- |
| Python from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler sc = StandardScaler() # MinMaxScaler or RobustScaler X\_train[numeric\_columns] = sc.fit\_transform(X\_train[numeric\_columns]) X\_test[numeric\_columns] = sc.transform(X\_test[numeric\_columns]) ##通过调用transform方法，可以使用前面获得的样本均值和方差来对数据做标准化处理 |

**Binning**

a b c

1

2

3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e |  |  |
| 1 | NA | 0.2 |  |  |  |  |  |
| 2 | 0.1 | 0.3 |  |  |  |  |  |
| 3 | 0.3 | 0.4 |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 1 |  | 0.2 |
| 2 | 0.1 | 0.3 |
| 3 | 0.3 | 0.4 |

0.2-0.4 equidistant 0.1

0.2 a

0.3 b

b1 b2 b3 b4

1 0 0

0 1 0

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 1 | A | 0.2 |
| 2 | b | 0.3 |
| 3 | b | 0.4 |

Numerical to categorical

Equidistant, Equal Frequency, Log and then take Integer.

Only perform log transformation on specific columns.

|  |
| --- |
| Python log\_cols = [\_ for \_ in X\_train.columns if \_ not in ['EJ', 'BN', 'CW', 'EL', 'GL']] X\_train.loc[:, log\_cols] = np.log1p(X\_train.loc[:, log\_cols]) X\_test.loc[:, log\_cols] = np.log1p(X\_test.loc[:, log\_cols]) |

**Unbalanced Data**

https://www.kaggle.com/competitions/icr-identify-age-related-conditions/discussion/412507

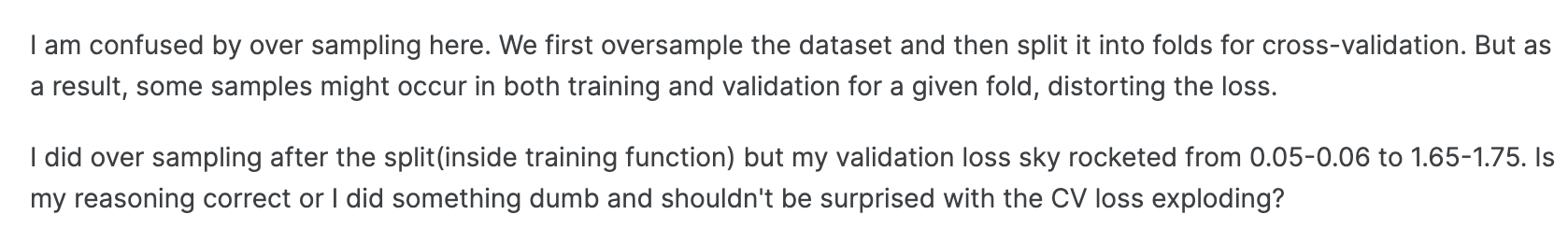
1.down/up sampling

|  |
| --- |
| Python def random\_under\_sampler(df): #if sampling\_method == 'under':  neg, pos = np.bincount(df['Class'])   one\_df = df.loc[df['Class'] == 1] #108  zero\_df = df.loc[df['Class'] == 0] #509    zero\_df = zero\_df.sample(n=pos) #108    undersampled\_df = pd.concat([zero\_df, one\_df])#216  return undersampled\_df.sample(frac = 1)   train\_good = random\_under\_sampler(train) |

Choose oversample

2. Oversampling, Data Synthesis

|  |
| --- |
| Python ## 2. oversampling ->  #SMOTE- generate new sample via interpolation  if sampling\_method == 'over':  X = train[selected\_cols]  y = train['Class']   smote = SMOTE(k\_neighbors=5)  # fit\_resample   X\_resampled, y\_resampled = smote.fit\_resample(X, y)  print(X\_resampled.shape, y\_resampled.shape)  X\_resampled["Class"] = y\_resampled  train = X\_resampled   #RandomOverSampler- Oversample by duplicating some of the original minority class samples  ros = RandomOverSampler(random\_state=42) train\_ros, y\_ros = ros.fit\_resample(train\_pred\_and\_time, greeks.Alpha) |



3. Solve this issue from the perspective of loss, look for some solutions like multitask MMOE.

Focal loss GHM loss

4. Class weight: scale\_pos\_weight can only be used for binary classification problems.

|  |
| --- |
| Python LGBMClassifier(class\_weight='balanced') XGBClassifier(scale\_pos\_weight=4.71) CatBoostClassifier(auto\_class\_weights='Balanced') LogisticRegression(class\_weight='balanced') LinearDiscriminantAnalysis(priors=[0.5, 0.5]) |

**Feature Selection**

1.vif:

[Use Variance Inflation Factor (VIF) for feature selection](http://sigmaquality.pl/models/feature-selection-techniques/feature-selection-techniques-variance-inflation-factor-vif-290320202006/)

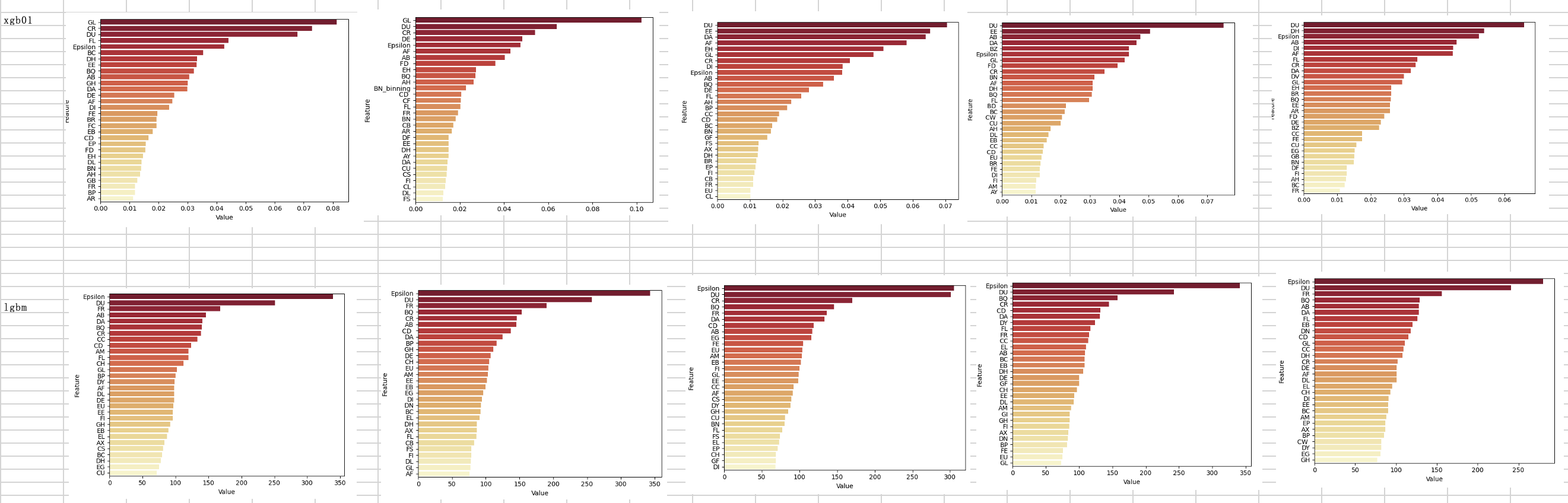
|  |
| --- |
| Python def check\_vif(df):  vifs = [variance\_inflation\_factor(df, i) for i in range(df.shape[1])]  vif\_df = pd.DataFrame({"features":df.columns, "VIF" : vifs})  vif\_df = vif\_df.sort\_values(by="VIF", ascending=False)  remove\_col = vif\_df.iloc[0, 0]  top\_vif = vif\_df.iloc[0, 1]  return vif\_df, remove\_col, top\_vif    # remove all features when VIF is over 10. if apply\_vif:  top\_vif = 100   while(top\_vif > 5):  vif\_df, remove\_col, top\_vif = check\_vif(train)  print(remove\_col, top\_vif)  if top\_vif < 5:  break  train = train.drop(columns=remove\_col)   display(train) |

1. Use Random Forest feature importance for selection.

|  |
| --- |
| Python X = train.drop(columns=["Class"]) y = train['Class']  if feature\_selection:  rf\_param\_grid = {'n\_estimators': 100, 'max\_depth': 10, 'max\_features': 0.7}  rf = RandomForestClassifier(random\_state=42, n\_jobs=-1)    rf.fit(X, y)  print("Train ACC : %.4f" % accuracy\_score(y, rf.predict(X)))  fi\_df = pd.DataFrame({'feature':X.columns, 'importance':**rf.feature\_importances\_**})  selected\_cols = fi\_df.sort\_values(by="importance", ascending=False)[:m]["feature"].values    display(selected\_cols)    X = train[selected\_cols]  display(X) |

1. XGB
2. EDA

EJ:column EJ is a redundant column in train data. All information of column EJ is contained in column EH. Whenever column EJ=A then EH=0.003042 and whenever column EJ=B then EH>=0.006084. So we can drop column EJ without losing any train data.



**Feature Crossing**

1.direct crossing

2.crossing after bucketing

**Feature importance**

xgb\_models.feature\_importances\_

**Xgb tuning**

|  |
| --- |
| Python import optuna import xgboost as xgb #trial.suggest\_categorical #trial.suggest\_float #trial.suggest\_int #binary:logistic #binary:logitraw #1. Define an objective function to be maximized. def objective(trial):  # 2. Suggest values of the hyperparameters using a trial object.  params = {  'n\_estimators' : trial.suggest\_int('n\_estimators',2000,3000),  'max\_depth': trial.suggest\_int('max\_depth',3,8),  'min\_child\_weight': trial.suggest\_float('min\_child\_weight', 2,4),  "learning\_rate" : trial.suggest\_float('learning\_rate',1e-4, 0.2),  'subsample': trial.suggest\_float('subsample', 0.2, 1),  'gamma': trial.suggest\_float("gamma", 1e-4, 1.0),  "colsample\_bytree" : trial.suggest\_float('colsample\_bytree',0.2,1),  "colsample\_bylevel" : trial.suggest\_float('colsample\_bylevel',0.2,1),  "colsample\_bynode" : trial.suggest\_float('colsample\_bynode',0.2,1),  }  xgbmodel\_optuna = XGBClassifier(\*\*params,random\_state=seed,tree\_method = "gpu\_hist",eval\_metric= "auc")  xgbmodel\_optuna.fit(X,y)  cv = cross\_val\_score(xgbmodel\_optuna, X, y, cv=4,scoring='neg\_log\_loss').mean()  return cv  # 3. Create a study object and optimize the objective function. study = optuna.create\_study(direction='maximize') study.optimize(objective, n\_trials=100,timeout=1200) |

**Post-processing**

[Find threshold](https://www.kaggle.com/code/aliasgherman/post-processing-thresholding-function-icr)

1. predict class



**Click the image to view the sheet.**

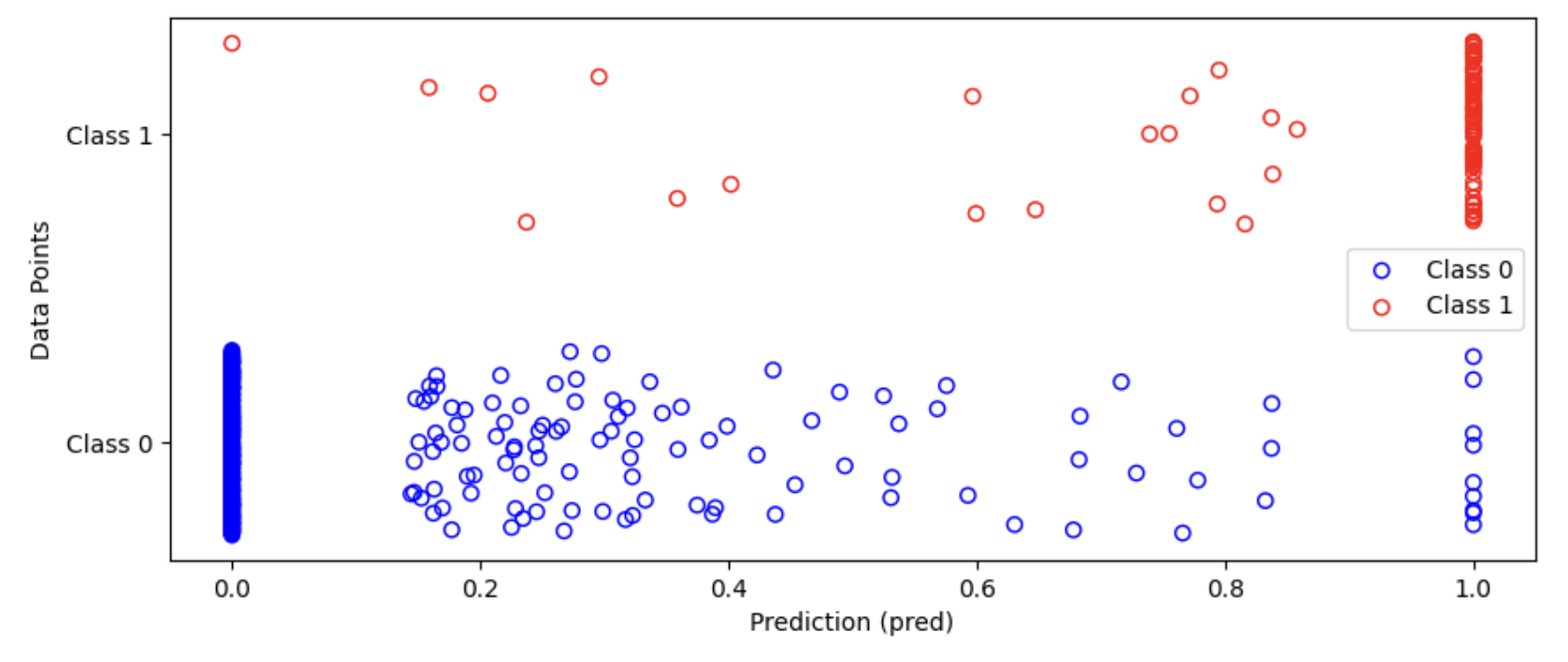
[use lb0.06 reference](https://www.kaggle.com/code/romanleo2003/postprocessing-ensemble)

1. 3xgb+lgbm ：

preds[preds > 0.86] = 1

preds[preds < 0.14] = 0

test\_local\_cv: [0.5655853642925102, 0.4071467407172298, 1.050574755207553, 0.3216936238663481, 0.4697278935147276]  
test\_Mean local\_cv: 0.5629456755196738

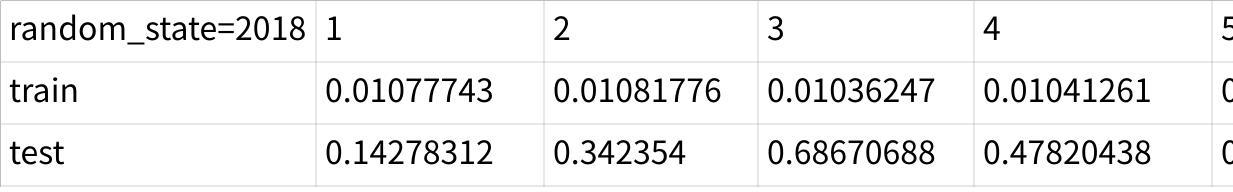


**Experiment result**

1. **7.3 base\_model xgb&lr**

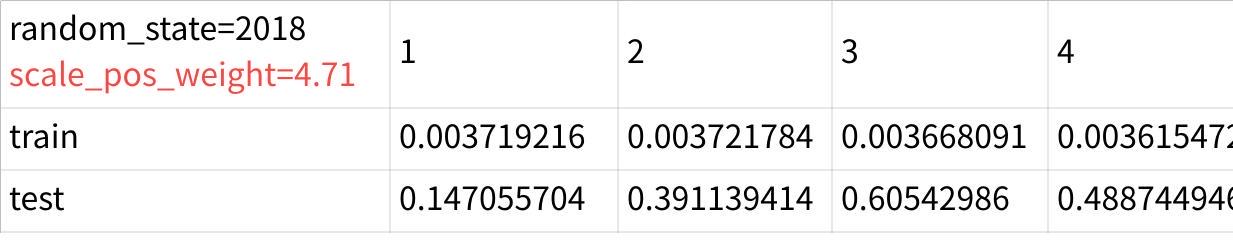
5-fold：random\_state=40

**Xgb**



**Click the image to view the sheet.**

7.6 add class weight

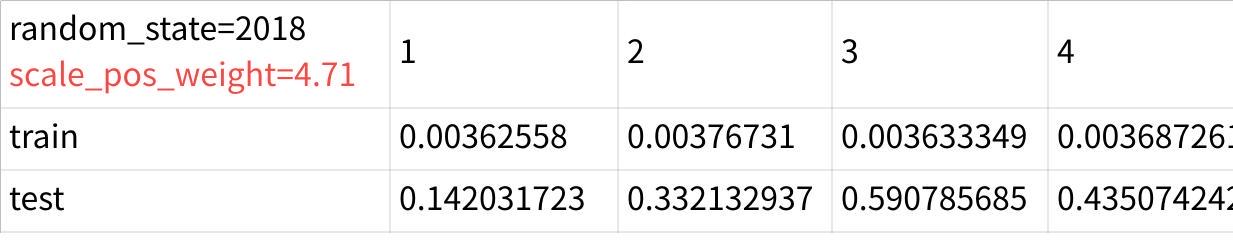


**Click the image to view the sheet.**

7.6 Concatenate timestamp (greeks data) and fill in missing time based on knn

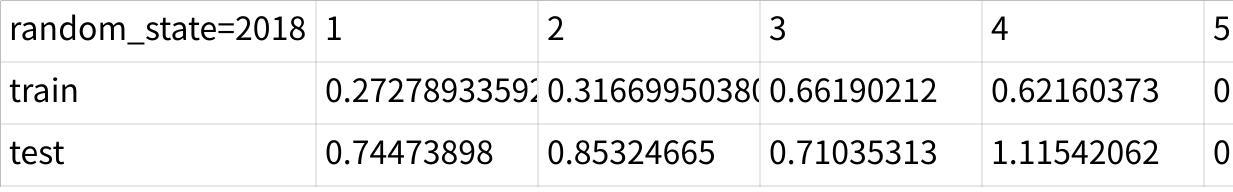


**Click the image to view the sheet.**



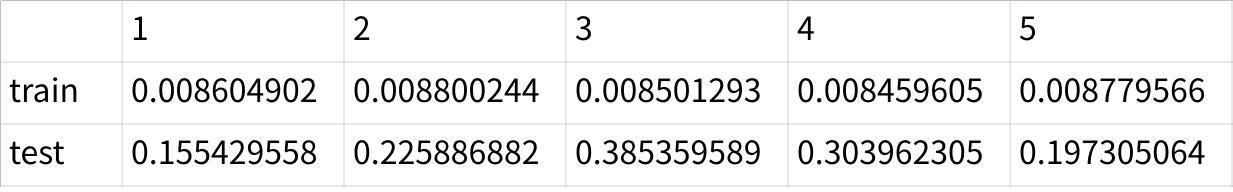
**Click the image to view the sheet.**

**LR**



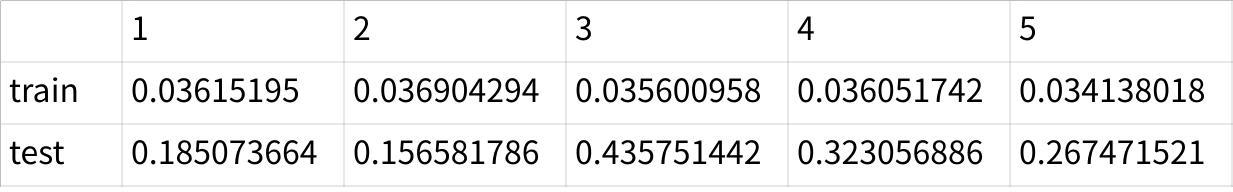
**Click the image to view the sheet.**

1. **7.7 base\_model xgb**



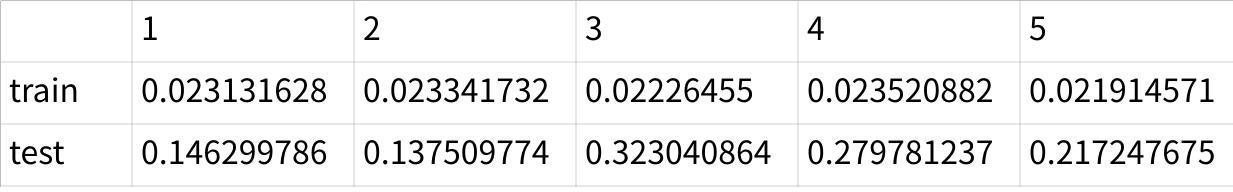
**Click the image to view the sheet.**

1. **7.7 base\_model TabPFN**



**Click the image to view the sheet.**

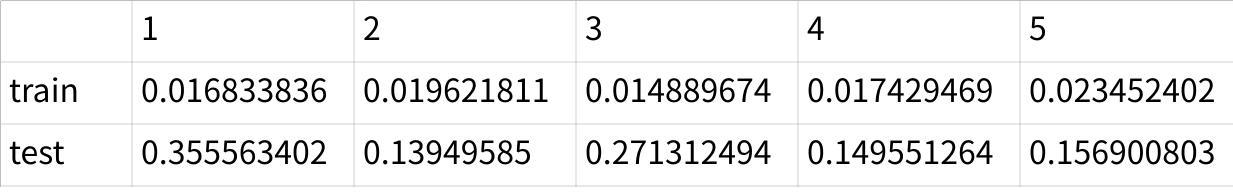
1. **0.5 xgb 0.5tabpfn**



**Click the image to view the sheet.**

1. **0.5 xgb 0.5tabpfn，greeks[alpha] as feature**

**But when calculating the loss, use the class, that is, add up the last three probabilities of the predicted alpha as the probability for class 1**



**Click the image to view the sheet.**

**Daily todo**

|  |  |  |
| --- | --- | --- |
| date | question | todo |
| 0708 | Xgb tuning; smote; regu; binning |  |
| 0709 | note | BQ col= age |
| 0720 | 1. Feature importance(including age) 2. Outlier deletion, cv score | 1. Prediction result visualization 2. BN w/o normalization 3. Cv 0.08 0.06 4. TabPFN—cv- tuning 5. Feature importance analysis 6. Discussion board |

[note](https://mxz9f3au4i.feishu.cn/sheets/Lrl6sL6pvhSFqttWyJPcrAsznFe)

A black text with purple lines

Description automatically generated

A screenshot of a graph

Description automatically generated

**Submitted**

|  |  |  |  |
| --- | --- | --- | --- |
| date | model | cv | lb |
| 0728 | 3xgb+lgbm+post（0.86，0.14)+oversample(外部）+预测alpha | 0.0312 | 0.16 |
| 0727 | 3xgb+lgbm+tabpfn+post（0.86，0.14)+pre\_class | 0.53 | 0.14 |
|  | 3xgb+lgbm+tabpfn+post（0.86，0.14)+pre\_class+oversample(inner） | 0.47 | 0.14 |
| 0726 | 3xgb+lgbm+pred\_class | **0.1685** | **0.17** |
|  | 3xgb+age\_col feature augmentation | 0.17 | 0.19 |
|  | 3xgb | 0.21 | 0.21 |
| 0731 | 3xgb+lgbm+ change **missing value imputation method + internal oversampling + pred\_class** | **0.1659** | **0.19** |
| 0801 | 3XGBoost models + LightGBM + Change missing value imputation method + internal oversampling + predict class + add column (whether BQ is empty); known that when BQ is empty, class is 0 | 0.1634 |  |
|  | 3 XGBoost models + LightGBM + Change missing value imputation method + internal oversampling + predict class + add column (whether BQ is empty; known that when BQ is empty, class is 0) + remove EJ. | **0.1580** | 0.19 |
|  | 3XGBoost models + LightGBM + TabNet + Change missing value imputation method + internal oversampling + predict class + add column (whether BQ is empty; known that when BQ is empty, class is 0) + remove EJ. | 0.1628 |  |

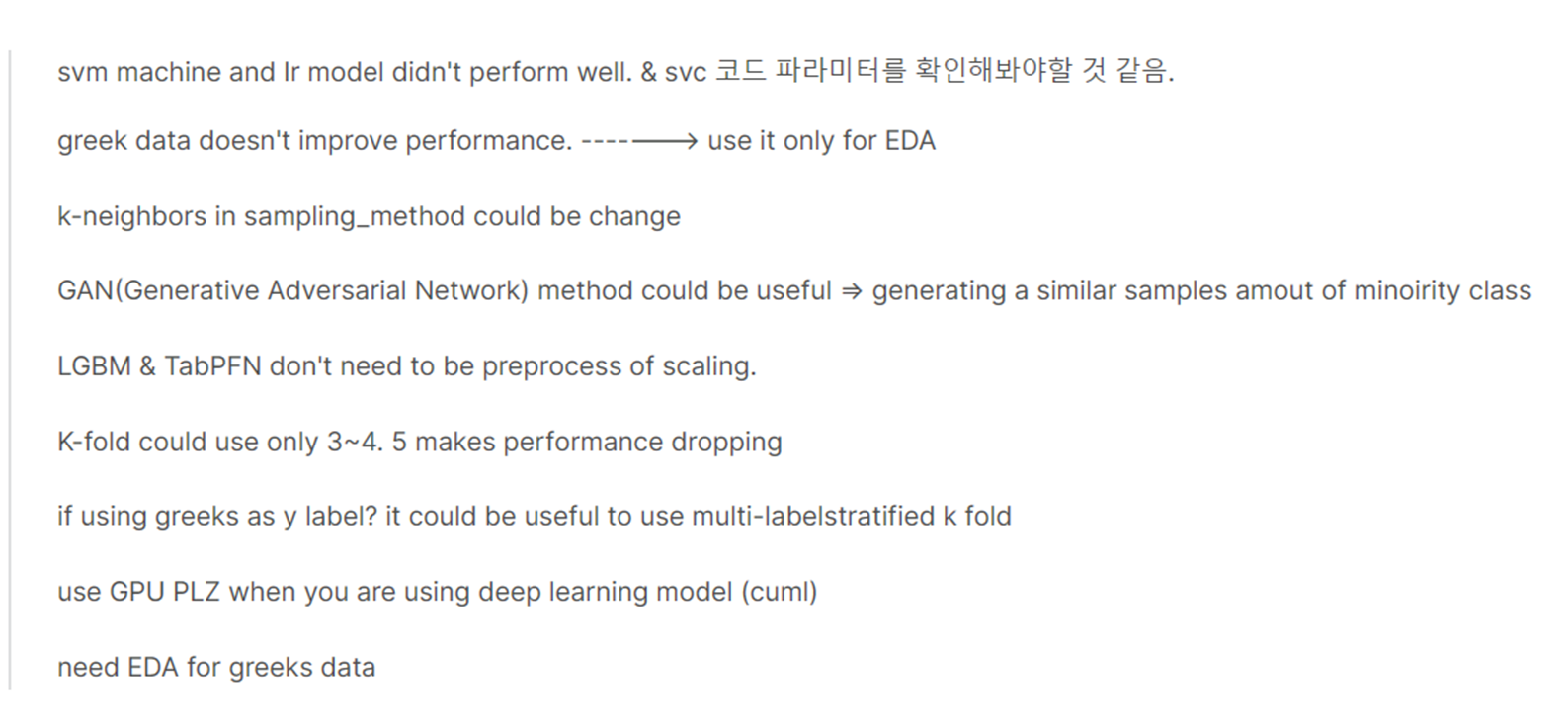
A graph of blue and red dots

Description automatically generated

**Unsubmitted local comparison**

|  |  |  |  |
| --- | --- | --- | --- |
| date | model | cv | comment |
| 0728 | 3xgb+lgbm+oversample(inner）+pred alpha | 0.1850 |  |
|  | 3xgb+lgbm+oversample(inner）+pred class | 0.1741 | **scale\_pos\_weight= 4.71** |
|  | 3xgb+lgbm+pred alpha | 0.1905 |  |
|  | 3xgb+lgbm+pred class | 0.1685 |  |
|  | 3xgb+lgbm+tabpfn+overwrite\_warning =True+pred class | 0.1769 |  |
|  | 3xgb+lgbm+tabpfn+pred class | 0.1769 |  |

**Useful Resource**



[Fork of SVC+RF+LR+XGB with Auto](https://www.kaggle.com/code/samuelpark97/fork-of-for-beginner-svc-rf-lr-xgbwith-auto)

https://www.kaggle.com/code/samuelpark97/fork-of-for-beginner-svc-rf-lr-xgbwith-auto

https://www.kaggle.com/code/moritzm00/icr-xgb-lgbm-pipeline-hyperparameter-tuning

